S&ZI0020104

Method of and Apparatus for Modular Multiplication

Cross-Reference to Related Application:

This application is a continuation of copending International Application No. PCT/EP02/00734, filed January 24, 2002, which designated the United States and was not published in English.

BACKGROUND OF THE INVENTION

1. Field of the Invention:

The present invention relates to cryptographic algorithms and apparatus for implementing such cryptographic algorithms, and in particular to a method of and an apparatus for modular multiplication using a multiplication look-ahead process and a reduction look-ahead process.

2. Description of the related art:

Cryptography is one of the essential applications of modular arithmetic. An essential algorithm for cryptography is the known RSA algorithm. The RSA algorithm is based on a modular exponentiation that can be represented as follows:

$$C = M^d \bmod (N),$$

wherein C is an encrypted message, M is a non-encrypted message, d is the secret key and N is the modulus. Modulus N usually is generated by multiplication of two prime numbers p and q. The modular exponentiation is broken down into multiplications by means of the known square-and-multiply algorithm. To this end, the exponent d is broken down into powers of two so that the modular exponentiation may be broken down into several modular multiplications. For being able to efficiently implement the modular exponentiation in terms of computation, the modular exponentiation thus is broken down into modular multiplications which may then be broken down into modular additions.

The document DE 3631992 C2 discloses a cryptographic process in which the modular multiplication can be accelerated using a multiplication look-ahead process and a reduction look-ahead process. The process described in DE 3631992 C2 is also referred to as ZDN method and will be explained in more detail by way of Fig. 9. After a start step 900 of the algorithm, the global variables M, C and N are initialized. The object consists in computing the following modular multiplication:

$$Z = M * C \bmod N.$$

M is the multiplier, whereas C is the multiplicand. Z is the result of the modular multiplication, whereas N is the modulus.

Then, there are various local variables initialized that need not be dealt with in more detail for the time being. Thereafter, two look-ahead processes are employed. In the multiplication look-ahead process GEN_MULT_LA, a multiplication shift value s_z as well as a multiplication look-ahead parameter a are calculated (910) employing various look-ahead rules. Following this, the current contents of the Z register are subjected to a left-shift operation by s_z digits (920).

Substantially parallel therewith, there is carried out a reduction look-ahead process GEN_Mod_LA (930) for calculating a reduction shift value S_N and a reduction parameter b. In a step 940, the current content of the modulus register, i.e. N, is shifted by S_N digits to the left or to the right, respectively, in order to produce a shifted modulus value N'.

The central three-operand operation of the ZDN method takes place in a step 950. In this step, the intermediate result Z' after step 920 is added to the multiplicand C that has been multiplied by the multiplication look-ahead parameter a, and to the shifted modulus N' that as been multiplied by the reduction look-ahead parameter b. Depending on the current situation, the look-ahead parameters a and b may have a value of +1, 0 or -1.

A typical case is that the multiplication look-ahead parameter a is +1 and that the reduction look-ahead parameter b is -1, so that the multiplicand C is added to a shifted intermediate result Z', and the shifted modulus N' is subtracted therefrom. a will have a value of 0 if the multiplication look-ahead process would allow more than a preset number of individual left-shifts, i.e. if s_z is greater than the maximum admissible value of s_z, which is also referred to as k. In the event that a is 0 and that Z', due to the preceding modular reduction, i.e. the preceding subtraction of the shifted modulus, still is quite small, in particular smaller than the shifted modulus N', no reduction has to take place to that the parameter b is 0.

Steps 910 to 950 are carried out until all digits of the multiplicand have been worked off or processed, i.e. until m is 0 and also until a parameter n is 0; this parameter indicates whether the shifted modulus N' still is greater than the original modulus N or whether, despite the fact that all digits of the multiplicand have already been worked off, still further reduction steps have to be carried out by subtraction of the modulus from Z.

Finally, it is determined whether Z is smaller than 0. If this is the case, it is necessary for achieving a final reduction that modulus N be added to Z so that the correct result Z of the modular multiplication is obtained in the end.

In a step 960, the modular multiplication by way of the ZDN method is concluded.

The multiplication shift value s_Z as well as the multiplication parameter a that are calculated in step 910 by the multiplication look-ahead algorithm, result from the topology of the multiplier as well as by the look-ahead rules employed which are described in DE 3631992 C2.

The reduction shift value S_N as well as the reduction parameter b, as described in DE 3631992 C2 as well, are determined by way of a comparison of the current contents of the Z register with a value 2/3 times N. This comparison gives the ZDN method its name (ZDN = Zwei Drittel N (= two thirds N)).

The ZDN method as illustrated in Fig. 9 returns the modular multiplication to a three-operand addition (block 950 in Fig. 9), in which the multiplication look-ahead process and, concomitantly therewith, the reduction look-ahead process, are employed for increasing computing time efficiency. Thus, an advantage in terms of computing time can be achieved in comparison with the Montgomery reduction.

In the following, the reduction look-ahead process performed in block 930 of Fig. 9 will be discussed in more detail by way of Fig. 10. Firstly, in a block 1000, a reservation is carried out for the local variables, i.e. the reduction look-ahead parameter b and the reduction shift value S_N. In a block 1010, the reduction shift value SN is initialized to zero. Then, the value ZDN is calculated in a block 1020, which is equal to 2/3 of modulus N. This value determined in block 1020 is stored in a register of its own, namely the ZDN register, in the crypto coprocessor.

It is then determined in a block 1030 whether the variable n is 0 or whether the shift value S_N is -k. k is a value that defines the maximum shift value preset by the hardware. In the first pass, block 1030 is answered NO such that in a block 1040, parameter n is decremented and that in a block 1060, the reduction shift value is decremented by 1 as well. In a block 1080, the variable ZDN then is allocated anew, namely with half of its value, which may easily by achieved by a right-shift of the value contained in the ZDN register. It is then determined in a block 1100 whether the absolute value of the current intermediate result is greater than the value contained in the ZDN register.

This comparison operation in block 1100 is the central operation of the reduction look-ahead process. If the question is answered YES, the iteration is terminated, and the reduction look-ahead parameter b will be allocated as shown in block 1120. If, in contrast thereto, the question to be answered in block 1100 is answered NO, the iteration jumps back in order to examine the current values of n and S_N in block 1030. If block 1030 is answered YES at any time in the iteration, the sequence jumps back to a block 1140 in which the reduction parameter b is set to zero. In the three-operand operation illustrated in block 950, this has the effect that no modulus is added or subtracted, which means that the intermediate result Z was so small that no modular reduction was necessary. In a block 1160, the variable n then is allocated anew, and in a block 1180 finally the reduction shift value S_N is computed which is required in a block 940 of Fig. 9 in order to perform the left-shift of the modulus so as to obtain a shifted modulus.

In blocks 1200, 1220 and 1240, the current values of n and k are finally examined with respect to further variables MAX and cur_k for examining the current allocation of the N register, in order to make sure that no register exceeding takes place. The closer details are not relevant to the present invention, but are described in detailed manner in DE 3631992 C2.

The algorithm shown in Figs. 9 and 10 can be implemented in terms of hardware as illustrated in Fig. 7. For the three-operand operation to be carried out in block 950, there is required an arithmetic unit 700, designated AU in Fig. 7. The latter is coupled with a register C 710 for the multiplicand, a register N 720 for the modulus and a register Z 730 for the current intermediate result of the modular multiplication. Fig. 7 reveals furthermore that the result of the three-operand operation, via a feedback arrow 740, is fed back to Z register 730. Fig. 7 illustrates furthermore the mutual connection of the registers. The value ZDN computed in block 1020 of Fig. 10 has to be stored in a ZDN register 750 of its own. The ZDN comparison, i.e. the iteration loop shown in Fig. 10, furthermore is controlled in its progress by a control logic 760 for the ZDN comparison of its own.

The main work of the ZDN algorithm for computing Z: = M x C mod N thus consists in the following two operations:

1. Computing the shift values s_z and s_i for the registers Z and N so as to fulfil the following equation:

   $2/3 \text{ N} \times 2^{-si} < |Z| \leq 4/3 \text{ N} \times 2^{-si}$ and

2. Computing the three-operand sum:

   $Z: = 2^{sZ} Z + a C + b \times 2^{sz-si} N,$

The multiplication look-ahead parameter a and the reduction look-ahead parameter b may assume values of -1, 0 and +1, as is known.

It is to be pointed out that the intermediate result Z, the multiplicand C and the modulus N are long numbers, i.e. numbers whose count of digits or bits may indeed be greater than 512, and which may also have up to more than 2048 digits.

The comparison of the current intermediate result Z with the value ZDN, which is to be carried out in block 1100, however, is not carried out for all bits of Z for reasons of computation time, but only with a number of most significant bits of Z; in this respect, a number of 32 bits has turned out to be sufficient for obtaining very high accuracy for the comparison result.

For the 32 most significant bits of 2/3 N required for this comparison, a register of its own is necessary which in Fig. 7 is indicated under reference numeral 750 and which is referred to as ZDN register.

Furthermore, a hardware comparator of its own is necessary which computes for the current value in the Z register and for the current value in the ZDN register the correct s_i value so that the following equation is fulfilled:

$$2/3\ 2^{-si}\ N < |Z| \leq 4/3\ 2^{-si}\ N$$

Thus, what is disadvantageous in this method is on the one hand that both the additional ZDN register and the hardware comparator require extra chip area. On the other hand, the computation of 2/3 N and the computation of the auxiliary shift value s_i in the ZDN algorithm performed by the iteration loop shown in Fig. 10 are time-critical for the entire algorithm and may indeed be determinative for the overall execution time of the algorithm.

Summary of the Invention

It is the object of the present invention to provide an improved concept for modular multiplication, which on the one hand can be implemented in more space-saving manner and on the other hand requires less computation time.

In accordance with a first aspect of the invention this object is achieved by a method of modular multiplication of a multiplicand (C) by a multiplier (M), in which a modulus (N) is employed, making use of a multiplication look-ahead process and a reduction look-ahead process, said method comprising the steps of:

transforming (10) the modulus (N) into a transformed modulus (N^T) that is greater than the modulus (N), with a predetermined fraction (2/3) of the transformed modulus having a higher-order digit with a first predetermined value that is followed by at least one lower-order digit having a second predetermined value;

iterative working off (12) of the modular multiplication making use of the multiplication look-ahead process and the reduction look-ahead process and utilizing the transformed modulus (N^T) so as to obtain at the end of the iteration a transformed result for the modular multiplication; and

re-transforming (14) the transformed result by modular reduction of the transformed result utilizing the modulus (N).

In accordance with a second aspect of the invention, this object is achieved by a processor for modular multiplication of a multiplicand by a multiplier, in which a modulus is employed, making use of a multiplication look-ahead process and a reduction look-ahead process, comprising:

a means for transforming the modulus into a transformed modulus that is greater than the modulus, with a predetermined fraction of the transformed modulus having a higher-order digit with a first predetermined value that is followed by at least one lower-order digit having a second predetermined value;

a means for iterative working off the modular multiplication making use of the multiplication look-ahead process and the reduction look-ahead process and utilizing the transformed modulus so as to obtain at the end of the iteration a transformed result for the modular multiplication; and

a means for re-transforming the transformed result by modular reduction of the transformed result utilizing the modulus.

The present invention is based on the finding that the comparison of the updated intermediate result with the value ZDN, i.e. 2/3 times modulus N, which comparison involves high expenditure in computation time, can be facilitated when the modulus N is first transformed into a transformed modulus N^T and the entire modular multiplication is carried out with the transformed modulus N_T instead of the modulus proper. According to the invention, the modulus is transformed such that the predetermined fraction of the transformed modulus, i.e. in a preferred embodiment, 2/3 times the transformed modulus, becomes a specific number that is selected such that the comparison of 2/3 N^T with the intermediate result Z becomes trivial. According to the present invention, the transformation is carried out such that the predetermined fraction of the transformed modulus has a higher-order digit with a first predetermined value, which is followed by at least one low-order digit having a second predetermined value. In binary representation and two's complement convention in which the most significant bit indicates the sign, the transformation of the modulus into a transformed modulus is carried out such that the second-most-significant bit of 2/3 N^T is a binary one, whereas the third-most-significant bit and still further less significant bits are zeroes.

In this event, the comparison is trivial such that it is simply necessary to count the number of the digits between the most significant one of the predetermined fraction of the transformed modulus and the updated intermediate result Z of the modular representation in order to obtain the shift value s_i from which the reduction shift value S_N can then be determined simply by subtracting the so-called auxiliary shift value s_i obtained by the ZDN comparison from the multiplication shift value of the multiplication look-ahead process taking place parallel thereto.

The entire ZDN operation is worked off exactly as in case of the prior art. However, instead of the modulus N, the transformed modulus N^T is employed, so that finally a "transformation result" of the modular multiplication is achieved which is in the remainder class of the transformed modulus N^T. A final re-transformation such that the transformation result of the modular multiplication is reduced in modular manner, making use of the original modulus N, will then yield the result proper of the modular multiplication of the multiplier M by the multiplicand C using modulus N.

Brief Description of the Drawings

Preferred embodiments of the present invention will be explained in detail hereinafter with reference to the accompanying drawings in which

Fig. 1 shows a flow chart of the concept for modular multiplication according to the invention;

Fig. 2 shows the splitting of a modulus N into a first section N^T of bits and into a second section N^R of bits;

Fig. 3 shows the splitting of the transformed modulus N^T into a first section of digits having a length L (N^T) and the remaining digits;

Fig. 4 shows a representation of the digits of 2/3 times the transformed modulus N^T;

Fig. 5 shows a schematic representation of the digits of the transformed modulus with randomization;

Fig. 6 shows a schematic representation of an arithmetic-logic unit for performing the modular multiplication according to the present invention;

Fig. 7 shows a schematic representation of an arithmetic-logic unit for the known ZDN method;

Figs. 8a to 8c show a schematic representation of the relationship between multiplication shift value s_z, auxiliary shift value s_i and reduction shift value S_N;

Fig. 9 shows a flow chart representation of the known ZDN method; and

Fig. 10 shows a flow chart representation of the known reduction look-ahead process.

Detailed Description of Preferred Embodiments

Fig. 1 shows a flow chart of the method according to the invention for modular multiplication of a multiplicand C by a multiplier M using a modulus N. At first, the modulus N is transformed, in a step 10, into a transformed modulus N^T in accordance with the following equation:

$$N^T = T \times N.$$

In a step 12, the modular multiplication is then worked off using the transformed modulus N^T and the predetermined fraction of the transformed modulus which is 2/3 in the preferred embodiment. With respect to a modular exponentiation, this means that an RSA equation of the following form is computed:

$$C^T: = M^d \bmod N^T.$$

Thus, the result of the modular exponentiation C is not computed in the remainder class defined by modulus N, but in the remainder class defined by the transformed modulus N^T, so that C^T, and not C, stands on the left side of the above equation. The concept according to the invention distinguishes itself in that, due to utilization of the transformed modulus N^T, the computation of the auxiliary reduction shift value s_i, which corresponds to the iteration loop of Fig. 10 of the known reduction look-ahead process, is highly simplified.

In a final step 14, a re-transformation of N^T into N is performed again, by carrying out an operation corresponding to the following equation:

$$C: = C^T \bmod N.$$

In this respect, the transformed result C^T that is in the remainder class of the transformed modulus N^T is returned to the remainder class of modulus N preferably by a simple shift/subtraction reduction, so that C is the result of the modular exponentiation.

The transformation of modulus N into a transformed modulus N^T utilizing the transformer T of step 10, is carried out such that the predetermined fraction of the transformed modulus, i.e. in the preferred embodiment, 2/3 times the transformed modulus, has a higher-order digit with a first predetermined value, which is followed by at least one low-order digit with a second predetermined value. The comparison of the intermediate result Z with 2/3 times the transformed modulus may thus be highly simplified, namely in that the uppermost digit of Z, which has the predetermined value as well, is looked for and the difference between the higher-order digit with the first predetermined value of the predetermined fraction of the transformed modulus and the uppermost digit of the intermediate result Z with the first predetermined value equals the difference s_i.

In summary, this can be represented as follows. N is transformed into a transformed modulus N^T preferably in the 32 bit CPU and not in the crypto coprocessor, so that the following holds:

$$N^T: = T \times N,$$

wherein T is a natural number.

The following form results for N^T, if all numbers used are binary numbers:

$$N^T = 1100\ldots\ 0\ XX\ldots XX$$

For 2/3 times the transformed modulus, the following value then results:

$$2/3\ N^T = 100\ldots\ 0\ X'X'\ldots X'X'$$

It can be seen from N^T and 2/3 N^T that both have a first share of e.g. 16 bits and then a share of L(N) bits X and X', respectively. For the so-called ZDN comparison, only the uppermost 16 bits of 2/3 times the transformed modulus N^T are utilized, since this already yields an error probability of better than approx. 2^{-10}. Thus, it is not necessary to use all 512, 1024 or 2048 bits of 2/3 times the transformed modulus for the ZDN comparison, but rather it is sufficient to perform this comparison with the uppermost 16 bits of the transformed modulus. Of course, it would be possible as well to use still fewer bits of 2/3 N^T for the comparison, but then the error probability increases gradually. However, as the errors are non-critical and result in sub-optimum behavior of the reduction look-ahead process only, this approach indeed is easily feasible.

2/3 times the transformed modulus N^T thus has a higher-order digit with the value 1, which is followed by at least one low-order digit with a value 0 and thus a second predetermined value. In the embodiment described hereinbefore, the number of the low-order digits is 15. It is of course possible here too to make use of higher or lesser numbers, depending on what dimensional differences are to be expected or handled between the intermediate result Z and 2/3 times the transformed modulus N^T. For the value of the intermediate result Z of the modular multiplication, i.e. the result of the three-operand addition in block 950 in Fig. 9, the following form results:

$$|Z| = 00...01YY...Y$$

The auxiliary shift value s_i is computed according to the following equation:

$$2/3\ N^T \times 2^{-si} < |Z| \le 4/3\ N^T \times 2^{-si}.$$

On the basis of the topology of 2/3 times the transformed modulus N^T, the value s_i always is the distance between the most significant bit with a 1 of 2/3 times the transformed modulus N^T and the most significant 1 of the value of the intermediate result.

According to the invention, this difference in digits or the value s_i can be determined in trivial manner. An iteration is no longer required.

In addition thereto, a ZDN register is no longer necessary for storing 2/3 times the modulus since, per definition, at least the upper e.g. 16 bits of 2/3 times the transformed modulus N^T always have the same form. A bit comparator is not necessary any more. The difference in significance between the highest-order digit of 2/3 times the transformed modulus N^T with a "1" and the highest-order digit of Z with a "1" can easily be established, for example, by a bitwise XOR operation of the register for the transformed modulus and the register for the intermediate result Z. s_i then is equal to the difference in significance of the digit where the XOR operation outputs a first "1" and where the XOR operation outputs a second "1".

Due to the fact that no ZDN register and no ZDN comparator are necessary, the overall arithmetic-logic unit can be accommodated on lesser chip area.

In addition thereto, the crypto control part, i.e. the control logic for the ZDN comparison (760 in Fig. 7), is of lesser complexity since the complex iteration loop of Fig. 10 need not be carried out. Finally, the computation is faster so that the computation of the auxiliary shift value s_i does no longer lead to timing problems for the entire algorithm.

In the following, the transformation according to the invention will be discussed in more detail by way of Figs. 2 to 5.

As has already been pointed out, a substantial part of the ZDN algorithm consists in that the following equation is fulfilled:

$$2/3\ 2^{-si}\ N < |Z| \leq 4/3\ 2^{-si}\ N.$$

s_i is referred to as auxiliary shift value and is the shift value that is necessary for shifting Z, in terms of digits, to the same position as N. In the prior art, the computation of s_i required comparison operations of |Z| with 2/3 N.

According to the invention, the comparison with 2/3 is simplified by transforming the modulus N into the transformed modulus N^T, with the transformed modulus N^T being greater than N, before any modular operation is carried out with N. All computations modulo N^T are carried out thereafter. However, since the result of the computation has to be in the remainder class N, a final reduction with N is carried out according to the invention.

As illustrated in Fig. 2, N is assumed to be an integer with a length of N bits. Due to the fact that modulus N always is a positive integer, i.e. MSB = 0 in two's complement representation, the sign bit equals 0 and the second-most-significant bit (MSB -1) of modulus N always is 1. It is not necessary for the ZDN comparison to compare all bits of the modulus to all bits of the intermediate result, but rather, it is sufficient to use a number of m bits for the ZDN comparison. The most significant m bits of the modulus N define a first part of modulus N_T, whereas the remaining N-m bits of the modulus define a second part N_R of the modulus. In a preferred embodiment, m is 16. Higher or lower values of m, of course, are possible as well.

As can be seen in Fig. 3, the transformation is carried out such that the transformed modulus N^T is 16 bits longer than the original modulus of Fig. 2.

It is sufficient for the ZDN comparison to utilize the first 16 bits of N^T, with a preferred embodiment of the present invention making use of only 12 bits for the comparison, while the 4 least significant bits constitute a buffer for possible carries that may come from still less significant bits.

In that event, the probability of the comparison yielding a wrong result is less than 2^{-12}. If the comparison yields a wrong result, there is just produced a sub-optimum reduction shift value S_N, however, the result modulo N remains correct.

If the modulus is utilized in two's complement representation as in Fig. 2, modulus N can be broken down as follows:

$$N = 2^{n-m} N_T + N_R.$$

N now is transformed into N^T using the transformer T, with T being an appropriately selected integer, which is necessary for reasons of congruence. N^T should have the form illustrated in Fig. 3, i.e. the most significant bit (MSB) of N^T must be 0, since N^T should be a positive integer. As elucidated hereinafter, the second-most significant bit and the third-most-significant bit of the transformed modulus must be 1, whereas all other bits of the uppermost section of the transformed modulus N^T, which section bears reference numeral 33 in Fig. 3, should have a value of "0". For, in this case only is the result for 2/3 times N^T that the uppermost section of 2/3 times N^T, as shown in Fig. 4, has only one bit with a "1", whereas all other bits in this uppermost section 44 are "0" so that the already described trivial comparison for determining s_i can be carried out.

However, the computation of the transformed modulus N^T using the transformer T shall be discussed first with reference to Fig. 3. The following definition is to be assumed:

$$N^T = T\ N$$

$$= T(2^{n-m}\ N_T + N_R)$$

The following holds for transformer T:

$$T = \left| \frac{2^{p-2} + 2^{p-3}}{N_T} \right|$$

Using equation 17, the following results for the transformed modulus N^T:

$$N^T = \left| \frac{2^{p-2} + 2^{p-3}}{N_T} \right| (2^{n-m}N_T + N_R)$$

$$N^T = (2^{n+p-m-2} + 2^{n+p-m-3})\frac{N_T}{N_T} + (2^{p-2}+2^{p-3})\frac{N_R}{N_T}.$$

If, for example typical values are taken for p and m, i.e. when p equals 32 bits and m equals 16 bits, the following results for N^T:

$$N^T = 2^{n+14} + 2^{n+13} + N_R\ \frac{2^{p-2}+2^{p-3}}{N_T}.$$

It is to be pointed out that the computation of N^T is preferably carried out in the host CPU and not in the crypto coprocessor. The host CPU comprises a short-number arithmetic-logic unit, which however is sufficient for computing N^T. Due to the fact that T has to be an integer and the computations are carried out within the crypto coprocessor modulo N^T instead of modulo N, with N^T being greater than

N, only the first p-m equal 16 bits of N^T are relevant for the trivial ZDN comparison in order to compute the auxiliary shift value s_i. The other n bits of N^T may be any number, they are not relevant for the computation of the auxiliary shift value s_i, i.e. for the comparison with Z. However, all bits of the transformed modulus N^T, of course, are necessary for the three-operand addition which now, instead of using the shifted modulus, is carried out using the shifted transformed modulus.

As shown in Fig. 17, the transformer T is a 16 bit integer for the values chosen for m and p. The division necessary for computing T and for computing N^T, respectively, thus has to be carried out for the most significant 32 bits only and thus can be programmed rapidly and easily on the host CPU.

Fig. 4 shows 2/3 times the transformed modulus N^T. As the MSB-1 and MSB-2 of N^T are "1", as shown in Fig. 3, and since the following holds:

$$(11)_2 = (3)_{10} \text{ and } (2/3 \times 3)_2 = (2)_{10} = (10)_2,$$

a simple bit pattern results for 2/3 times the transformed modulus N^T, with the length of 2/3 times the transformed modulus N^T being n-m+p.

Due to the special form of 2/3 N^T, the comparison with |Z| now becomes very simple. It is known that the highest-order one of 2/3 N^T is at a position n+p-m-2 at the beginning of a modular operation. A pointer for the register Z in a preferred embodiment then starts at the MSM of Z and looks for the first "1" of Z. If the MSB of Z is 1, Z will be a negative number and, instead, the first zero of Z will be looked for.

The difference of the bit position of the first one in register N and in register Z determines the auxiliary shift value s_i.

Due to the fact that the result of the modulo operation has to be in the remainder class N, a final reduction modulo N is carried out according to the invention, which means that a re-transformation has to be carried out (step 14 in Fig. 1).

The transformation of N into N^T has the following advantages as compared to the known ZDN comparison:

Instead of computing 2/3 N within the crypto coprocessor, a simple transformation of N into N^T can be carried out in the host CPU.

There is no ZDN register and no comparator logic necessary on the chip, so that the chip size is reduced and the complexity of the coprocessor decreases.

Finally, the transformation of N into N^T may be combined with randomization of modulus N as illustrated by way of Fig. 5. When R is a random number having a length of s bits, the randomized transformed modulus N^T has the form shown in Fig. 5. Due to the randomization number N, the randomized transformed modulus, as compared to the case in which no randomization has been carried out (Fig. 3), becomes longer by s bits, i.e. by the number of digits of R.

In the form of an equation, this may be expressed as follows:

$$N^T = T\ N$$

$$= T(2^{n-m}\ N_T + N_R)$$

The randomized transformer T then is as follows:

$$T = \left| \frac{2^{p-2}-2^{p-3}+R}{N_T} \right|$$

Thus, the following expression results for the randomized transformed modulus:

$$N^T = \left| \frac{2^{p-2}+2^{p-3}+R}{N_T} \right| (2^{n-m}N_T+N_R)$$

$$N^T = (2^{n+p-m-2}+2^{n+p-m-3}+R\ 2^{n-m}) \frac{N_T}{N_T} + (2^{p-2}+2^{p-3}+R) \frac{N_R}{N_T}.$$

When selecting p to have 144 bits, m to have 16 bits and s to have 112 bits, the following value results for the transformed modulus N^T including randomization:

$$N^T = 2^{n+126} + 2^{n+125} + R\ 2^{n-16} + N_R \frac{2^{144}+2^{143}+R}{N_T}.$$

The bith length of N^T then is:

$$L(N^T) = n+p-m = n+m+s = n+16+112 = n+128 \text{ bits}$$

Fig. 6 illustrates an arithmetic-logic unit according to the invention which, as compared to Fig. 7, no longer has a ZDN register, but merely an arithmetic unit 700, a C register 710, an N register 720 and a Z register 730, with the N register 720 no longer storing the modulus or a shifted modulus, but the transformed modulus or a shifted transformed modulus, or a randomized transformed modulus or a shifted randomized transformed modulus.

In the following, Figs. 8a to 8c shall be dealt with in order to illustrate the relationship between the auxiliary shift value s_i and the reduction shift value S_N.

In the following, Figs. 8a to 8c shall be dealt with in order to illustrate the computation of the reduction shift value s_z using the auxiliary reduction shift value s_i. Fig. 8a shows an intermediate result Z and a modulus N. Merely by way of example, the intermediate result has 4 bits, while the modulus has 9 bits. It is to be assumed now that the block 214 of Fig. 2 computes a shifted intermediate result Z', which can be achieved by multiplication by s_z. Assuming that the multiplier had 8 zeroes, the result hereof is that the multiplication shift value $s_{z'}$ is 8. To obtain a modular reduction, the modulus N must be brought to the order of magnitude of the shifted intermediate result Z'. According to the invention, the modulus N is to be shifted to such an extent that the uppermost bit of the shifted intermediate result polynom Z' and the uppermost bit of the shifted modulus N are equal. As can be seen from Fig. 8b, a reduction shift value S_N of 3 is required in this respect.

It can also be seen from Fig. 8b that the determination of S_N actually can be carried out only after s_z has been computed, i.e. that parallel implementation of blocks 210 and 212 of Fig. 2, as is preferred for the present invention, is not possible. For this reason, the auxiliary shift parameter s_i is introduced. The advantageous aspect of s_i is that this value can be computed without the s_z of the current step being known.

It can be seen from Fig. 8b that s_Z at all time is equal to the sum of s_i and s_N. s_N thus is always correlated with s_Z and s_i such that the following equation holds:

$$s_N = s_Z - s_i.$$

The time-consuming iterative process for determining s_N thus can be broken down into a time-consuming iterative process for determining s_i (loop 416) and a fast difference operation (block 422 of Fig. 4). Thus, nearly parallel implementation of the two look-ahead processes is possible, with the sole serial component consisting in that, prior to computing block 422 (Fig. 4), the actual value of s_Z has already been computed and delivered by the multiplication look-ahead algorithm (arrow 230 in Fig. 2).

It is to be summarized that the present invention simplifies the comparison between 2/3 N and the value of Z as compared to the known ZDN method. In contrast to the method known so far, in which the uppermost 32 bits of 2/3 N were computed in the crypto coprocessor and deposited in a separate 32 bit register, the ZDN register, with the comparison of 2/3 N with the value of Z having been carried out according to the known ZDN method in hardware via a comparator that was constituent part of the control part of the crypto coprocessor, the method now proceeds as follows. The modulus N is transformed by the host CPU into a transformed modulus N^T that is greater than N, with the first bits of N^T being a constant that is selected such that the comparison of 2/3 N^T with the value of Z is trivial. For improving security against information leakage attacks, such as SPA, DPA, timing attacks, ..., the transformation of N into N^T may be combined with the randomization of the modulus, as has been illustrated.

The 2/3 N computation in the crypto coprocessor is thus dispensed with. The ZDN register and the comparator logic are omitted as well, thus providing for smaller chip area and reduction of the complexity of the control part in the crypto coprocessor by omission of the comparator logic.